USING BLED112 WITH RASPBERRY PI

QUICK START GUIDE

Tuesday, 15 April 2014

Version 1.1

bluegiga

VERSION HISTORY

Version	Comment
1.0	First version
1.1	Minor changes

TABLE OF CONTENTS

1 Introduction

This guide contains the basic setup instructions needed to start using the BLED112 *Bluetooth* Smart USB dongle with the Raspberry Pi. **Note that some general Linux experience is assumed and will greatly help with development or troubleshooting in this process.**

Figure 1: BLED112 dongle

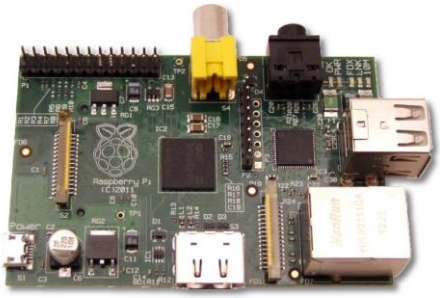

Figure 2: Raspberry Pi Model B

Since the BLED112 is a simple USB dongle, the only action necessary is to plug it into one of the Raspberry Pi's available USB ports. This may be done at any time, before or after the system has power or has finished booting.

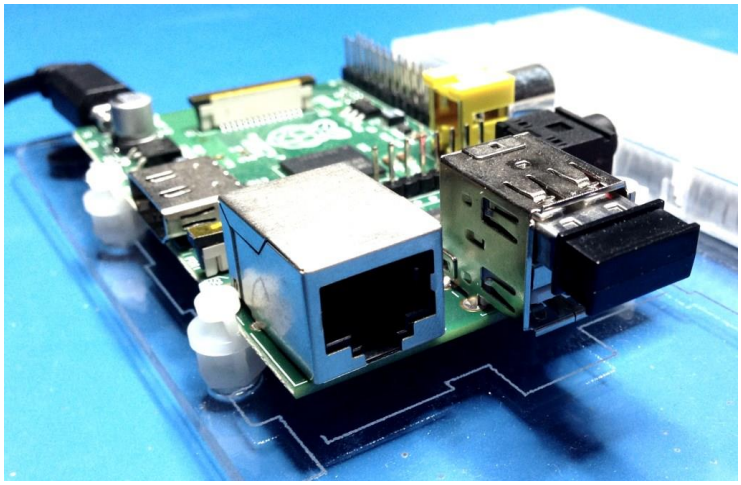

Figure 3: BLED112 in a Raspberry Pi USB port

2 Setting up Software

Use the current standard Raspbian “Wheezy” Linux build available here:

- http://www.raspberrypi.org/downloads

No drivers should be necessary, since the BLED112 is a full-stack device which enumerates as a USB CDC port. It does not require any Bluetooth stack such as “bluez” to be present on the host. However, you can still verify device connectivity and that the correct kernel module has been loaded by connecting to the Raspberry Pi via SSH (or a direct terminal session with a keyboard and display), and then checking the output of “**`lsusb`**”, “**`lsmod`**”, and “**`ls /dev/ttyACM*`**” as shown here:

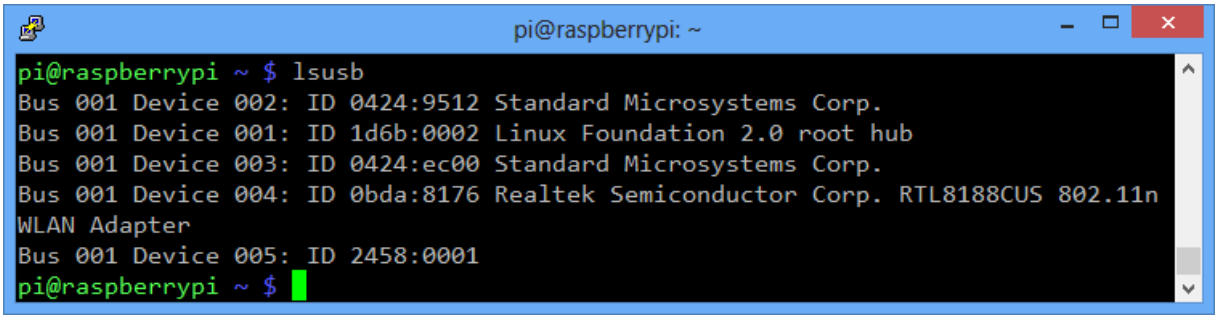

Figure 4: “`lsusb`” output (BLED112 appears as **`ID 2458:0001`**)

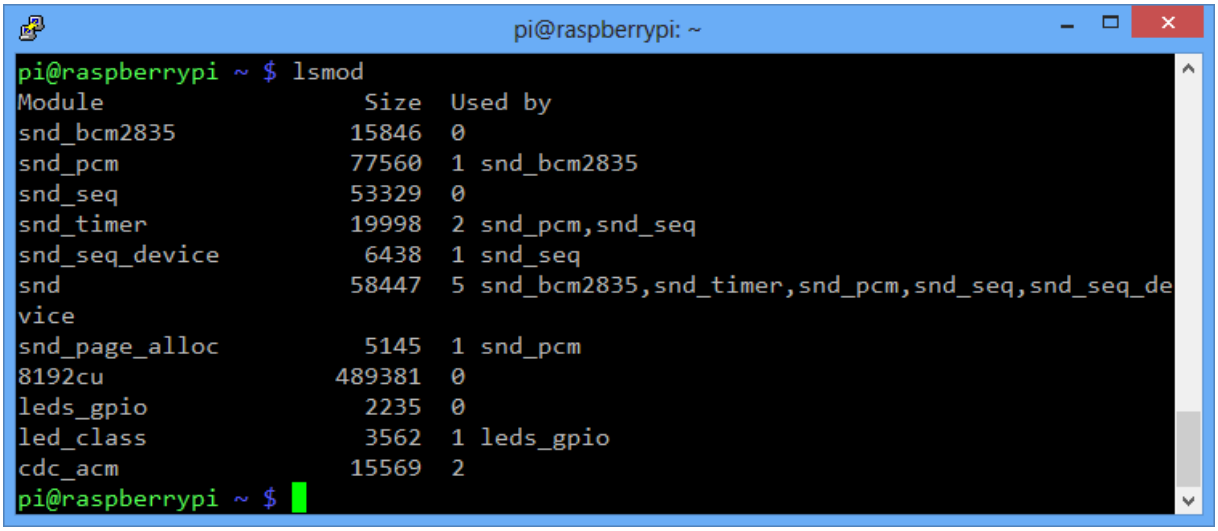

Figure 5: “`lsmod`” output (BLED112 uses **`cdc_acm`** module)

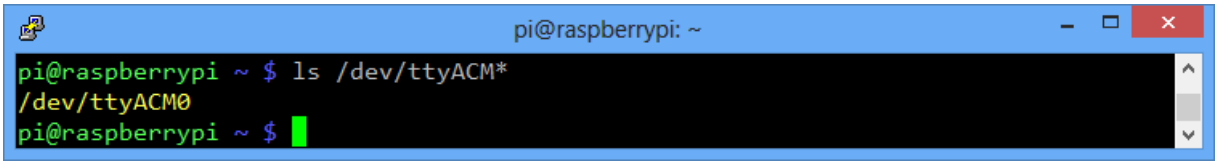

Figure 6: “`ls /dev/ttyACM*`” output (BLED112 appears as **`/dev/ttyACM0`**)

2.1 Testing *Bluetooth* Smart communications

Testing *Bluetooth* Smart communication:

This guide was written with the following starting configuration:

- Raspberry Pi model B, 256MB version
- Fresh 4GB SD card written with **2013-02-09-wheezy-raspbian.img**
- Bluegiga BLED112 running factory default "**usbcdc**" firmware
- Wired Ethernet connection to a local network connected to the internet
- Any Bluetooth Smart peripheral device nearby

Your configuration may be slightly different, but the instructions here should still apply. To prepare and run a basic Python-based BLE scanner, enter the following commands from a terminal or SSH session:

```
sudo apt-get install python-serial
wget https://raw.github.com/jrowberg/bglib/master/Python/Examples/bled112_scanner.py
chmod +x ./bled112_scanner.py
./bled112_scanner.py
```

This will install the **python-serial** package (which provides PySerial), then download the Python BGAPI-based BLE scanner script, make it executable, and finally run it using all of the default parameters. If there are any advertising BLE peripheral devices nearby, then you should begin seeing output like the following:

```
============================================================
BLED112 Scanner for Python v2013-05-10

============================================================

Serial port:    /dev/ttyACM0
Baud rate:      115200
Scan interval:  200 (125.00 ms)
Scan window:    200 (125.00 ms)
Scan type:      Passive
UUID filters:   None
MAC filter(s):  None
RSSI filter:    None
Display fields: - Time
                - RSSI
                - Packet type
                - Sender MAC
                - Address type
                - Bond status
                - Payload data
Friendly mode:  Disabled
------------------------------------------------------------
Starting scan for BLE advertisements...
1368200810.904 -52 0 000780535BB4 0 255 020106020A0306FFFFFFB1B2B3
1368200812.410 -52 0 000780535BB4 0 255 020106020A0306FFFFFFB1B2B3
1368200813.915 -52 0 000780535BB4 0 255 020106020A0306FFFFFFB1B2B3
```

You can explore the various argument options for this scanner script by running it with the **-h** or **--help** argument.

Note that if you do not have a BLE peripheral device nearby but you do have an iPhone 4S+ or iPad 3+ or iPad Mini running iOS 6+, then you can use some freely available iOS apps to emulate a heart rate sensor or temperature sensor peripheral device.

3 Testing more complex communication with BGAPI

The **bled112_scanner.py** script implements only a limited subset of the full BGAPI communication protocol, but you can do a lot more with BGAPI than just scan for other devices. You can act as a BLE master (central/manager) device and connect to other peripherals and use them, or you could even act as a BLE peripheral device to allow a BLE master such as an iPhone or iPad to connect and control the Raspberry Pi.

To get started with something more complicated, you can download the Python **bglib** library, which is a port of the C wrapper we provide in our SDK. To download this code and a couple of example scripts which use it, enter the following commands in a terminal or SSH session:

```
wget https://raw.github.com/jrowberg/bglib/master/Python/bglib.py
wget https://raw.github.com/jrowberg/bglib/master/Python/Examples/bglib_test_scanner.py
wget https://raw.github.com/jrowberg/bglib/master/Python/Examples/bglib_test_htm_collector.py
wget https://raw.github.com/jrowberg/bglib/master/Python/Examples/bglib_test_hr_collector.py
chmod +x ./bglib_test_*
```

Note that you will still need to install the **python-serial** package as shown in the earlier scanner example if you do not already have it.

Each of the three example scripts above are built around the event-driven **bglib** Python library. The library itself is a single-file Python module which can be imported into any Python application. The examples above have the following functionality:

- **bglib_test_scanner.py:**
  Similar to the “bled112_scanner.py” example, this application uses the command/ response/event structure of BGAPI to scan for nearby BLE devices and display any resulting advertisement packets.

- **bglib_test_htm_collector.py:**
  This application scans for BLE devices which are advertising the official “Health Thermometer” service, then automatically connects when it finds one and configures the remote device to send temperature readings. These readings are displayed as they come in.

- **bglib_test_hr_collector.py:**
  This application scans for BLE devices which are advertising the official “Heart Rate” service, then automatically connects when it finds one and configures the remote device to send heart rate measurements. These are displayed as they come in.

4 Additional information

The latest data sheets, design references, and full API Reference Guide are available in Bluegiga’s BLED112 product page:

- http://www.bluegiga.com/en-US/products/bluetooth-4.0-modules/bled112-bluetooth-smart-dongle/documentation/

Always refer to the latest documentation when working with the BLED112 *Bluetooth* dongle.

5 Support

Technical support is available online: http://www.bluegiga.com/support